DOCUMENT RESUME

ED 295 668 IR 013 379

AUTHOR Tessmer, Martin; Jonassen, David H.
TITLE A CBI Model for the Design of CAI Software by Teachers/Nonprogrammers.
PUB DATE Jan 88
NOTE 15p.; In: Proceedings of Selected Research Papers presented at the Annual Meeting of the Association for Educational Communications and Technology (New Orleans, LA, January 14-19, 1988). For the complete proceedings, see IR 013 331.
PUB TYPE Speeches/Conference Papers (150) -- Reports - Evaluative/Feasibility (142) -- Guides - Classroom Use - Guides (For Teachers) (052)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Courseware; *Instructional Design; *Models; Programing; *Screens (Displays); *Teacher Developed Materials; Textbooks
IDENTIFIERS Screen Format

ABSTRACT

This paper describes a design model presented in workbook form which is intended to facilitate computer-assisted instruction (CAI) software design by teachers who do not have programming experience. Presentation of the model is preceded by a number of assumptions that underlie the instructional content and methods of the textbook. It is argued that teachers will and should continue to design their own software, but should not have to learn programming in order to do so. Thus, teachers need to be taught only a few basic instructional events that suit the learning outcome and CAI strategy they use, and these basics can be taught through a tutorial workbook approach. It is suggested that, to be effective, teacher-designed CAI should be outcome-based, should include an introductory readiness for learning microstrategy, and can utilize off-screen activities. The seven-part instructional program used to teach the CAI model includes: (1) determination of the objective of the lesson; (2) a task analysis of the lesson; (3) classification of the learning outcomes of the lesson; (4) design of a readiness for learning strategy; (5) design of the CAI instructional component; (6) conversion of workbook program design into screen displays; and (7) a formative evaluation of the lesson via screen displays. Finally, the CAI model is briefly evaluated based on its use in a graduate education course at the University of Colorado at Denver. The text is supplemented by one figure. (8 references) (EW)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED295668

Martin Tessmer & David Jonassen

AECT, 1988

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

Minor changes have been made to improve reproduction quality.

Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

A CBI MODEL FOR THE DESIGN OF CAI SOFTWARE BY TEACHERS/NONPROGRAMMERS

IR013379

BEST COPY AVAILABLE

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Michael Simonson

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

INTRODUCTION

This presentation describes the assumptions and methods of a CAI design process that teachers can use to design CAI software. The design model is intended to facilitate software design by teachers who may not know how to program a computer, but the model can be used by teacher-programmers as well. The model is intended for teachers at all levels, elementary, secondary, and postsecondary. The model has been field tested with secondary and postsecondary instructors as a textbook (Tessmer, Caverly, & Jonassen, in press) in the graduate teacher education program at the University of Colorado at Denver.

The textbook is a workbook that guides students through the design of a CAI lesson on paper. In this presentation, we would like to explain three components of this design model exemplified in this text: 1) the assumptions that underlie the instructional content and methods of the text, 2) the instructional methods themselves and 3) the results of our use of the workbook in a teacher training class.

To begin with, we would like to define several terms that will be used in this presentation:

1. "CAI Strategy". When we speak of a CAI strategy, we are speaking of ways a teacher can use the computer to facilitate learning of a given learning outcome. These methods include tutorials, drill and practice, simulations, games, and problem solving tools such as database management tools. These strategies may or may not involve stand-alone CAI instruction.

2. "Programming". We believe there are two types of CAI programming:

   a) programming as coding, the more common definition of programming, involves entering the program commands into the computer,

   b) programming as content and structure design, involves specifying the lesson content and structure in screen displays. In this second type of programming, the CAI program is created.

As we shall see, we assume that these programming types are not only conceptually but realistically distinct. Inotherwords, both types of programming can be done by the same or separate people for the same CAI lesson.

ASSUMPTIONS

The Role of Teachers in Software Design and Production

1. Teachers will continue to produce their own computer software. The quality of commercial CAI software is increasingly better, but teachers will still produce their own. This is because:

   a) as with all media, some teachers prefer to "roll their own" regardless of the quality of available software and the time it takes.

   b) there is still a large amount of poor quality software for many content areas (Schiffman, 1986; Smith, 1987).

   c) computers are increasingly available to teachers at all levels of schooling.

   d) more districtwide support is available for teachers to design software.

   e) more teachers are becoming comfortable with computers and thus are willing to try developing CAI.

   f) programming help for software development is increasingly available to teachers.

   g) When teachers must code CAI themselves, the latest authoring systems (e.g. Course of Action) make it increasingly easier to code a CAI lesson.

2. Teachers should not have to code a computer to program it, they program it through the design of screen displays. Better software is produced when the teacher can spend time on the design of the CAI program on paper screen displays, which then can be given to a programmer for program coding. This allows the teacher to concentrate on the content and teaching methods of the program

(although programming knowledge is helpful). Screen displays are job aids that help teachers format the information properly. As Hazen (1985) indicates, good formatting has motivational as well as instructional properties. More important, screen displays guide the teachers into a frame-based approach to software design, an approach that is effective for a variety of CAI strategies (Bonner, 1987). Most important, using screen displays forces the teacher to design the program on paper before any coding is done, so that they pay attention to the content and methods of the instruction, improving the instructional validity of the product.

3. In many school settings, teachers will function as the designers and subject matter experts of the program, but do not have to function as the program coders, and should not. Several CAI designers recognize that the ideal software production team in education consists of a designer, teacher, and programmer (Collis & Gore, 1987; Smith, 1987). However, instructional designers are not readily available to most teachers, particularly at the elementary and secondary school level. On the other hand, programmers are becoming more prevalent, particularly at the secondary level, where students are more proficient programmers.

4. In most school settings, instructional software will be utilized in situations where the teacher, students, media, and other resources are available to supplement the CAI lesson; the software does not usually function as a stand-alone lesson. As Schiffman (1986) has indicated, teachers incorporate software within an overall teacher-designed lesson plan that can include a variety of off-screen activities in the lesson. These activities involve peer

consultation, worksheets, practice and testing, and teacher help.

Methods for Learning Software Design

1. Teachers best learn how to "program" a computer through a tutorial workbook approach. This approach allows them to gradually plan their lesson through workbook exercises in each chapter of the text, and then convert the exercises directly into screen displays for final coding. A workbook requires the student immediately applies what they have learned in a chapter to their own project, avoiding the mistake of the "now let's put it all together approaches" of books that merely describe CAI. These books require the novice to synthesize all that they gave learned at the very end of the book, or they just offer descriptive advice with no design activities whatsoever. Also in a workbook, teachers first devote their efforts exclusively to the design of the content and methods of the program and not to how to code it or even design it on screen displays. After teachers have designed a program once or twice in this manner, they can then use screen displays immediately for lesson design.

2. To design effective CAI, teachers need to learn a few basic instructional events that uniquely suit the both the learning outcome and CAI strategy they use. As Resnick (1976) has argued, learners can elaborate upon a skilled performance routine once they have mastered a basic routine. Although they are usually novice programmers, teachers have a repertoire of teaching techniques and strategies that they use in instruction. What they need to know are the essential instructional events for the given learning outcome and CAI strategy they teach. They will embellish these with their

own instructional events as they design the program. The mistake of many CAI design books is they try to teach too many design options and factors for a single type of approach: they overload the novice with detail and options, depending on the novice to "figure out" what they need for their lesson, or for a teacher to teach them. These books often do not allow the student to construct anything as part of the text activities, nor do they furnish any programming guidelines via job aids.

3. Teachers should learn the general types of CAI strategies and the types of learning outcomes, which can be used to limit the range of possible CAI lesson features to those that suit both the strategy and outcome. Teachers learn to apply this limited set of design options to create a CAI lesson, and to embellish them with their own teaching methods.

Program Characteristics of Effective CAI

1. Effective CAI design is outcome-based. The key to the teacher's selection of apropriate CAI lesson characteristics lies in their classification of the learning outcome(s) of that lesson. The learning outcome of a CAI lesson helps determine the lesson objectives, learner readiness activities, instructional content and methods, practice and test items, and formative evluation criteria. To classify outcomes, teachers learn to write objectives, conduct simple task analyses, and describe the characteristics of different learning outcomes.

2.Effective CAI lessons embody a microstrategy (no pun intended) composed of instructional events that suit both the learning outcome and general CAI strategy selected by the teacher. There are five

basic types of CAI strategies (deductive/inductive tutorial, game, simulation, drill/practice, problem solving tools). Each has a different set of instructional events that are embodied in the program, depending upon the learning outcome of the instruction (concept learning, rule/principle learning, verbal information, problem solving). This actually creates a 5 x 4 matrix of types of CAI (Figure 1). This matrix approach elaborates upon the Gagne, Wager, and Rojas approach (1981), in that the CAI designer selects instructional events not only by learning outcomes, but also by CAI strategy.

Regardless of the CAI strategy used, the outcome-based performance remains the same for a given learning outcome (i.e., whether it is a simulation or tutorial, example classification is a requisite student performance in the lesson). However, the instructional events vary for the same outcome depending on the strategy ("eliciting the response " differs in a concept game vs. problem solving tool vs inductive tutorial). In some cases, an event may not be present at all (no "present the information" event in a game or problem solving tool). Teachers then learn lesson design best by learning how to: a) identify the outcomes of their lesson (the most crucial step), b) select a CAI strategy for that outcome, and c) utilize the instructional events of that strategy/outcome microstrategy to design the instructional component of the lesson.

3. Every CAI lesson should contain an introductory Readiness for Learning microstrategy that encompasses two events: a) recall of prior knowledge, b) preparation for new knowledge. These elements

are distinct in their definition and their effects on learning. They each can be done by the computer or via off-screen activities. They are at once the two most neglected but important components of a CAI lesson.

4. Effective CAI lessons can utilize off-screen activities as part of their microstrategy events. In many cases, asking the student to consult a book, teacher, or student may be a viable part of the computer lesson (especially for problem-solving outcomes that have practice and evaluation components that are difficult to design for a novice programmer). In particular, research has indicated that diadic and triadic student groupings can facilitate learning and performance in problem solving activities. For example, readiness activities can be conducted by the instructor before students start on a lesson. This CAI design assumption is particularly crucial to CAI designers that are teachers, for two reasons: a) experienced instructors can utilize lesson activities that they know have worked, even if they cannot be programmed into the computer, b) CAI lessons with off-screen activities can be easier to design and code than a similar stand alone lesson that programs all activities into the computer. Once off-screen activities are legitimized as part of the overall CAI process, teachers lose the common and harmful presupposition that CAI instruction must all be in the computer as a stand-alone process; that using media, text, or people is "cheating".

THE INSTRUCTIONAL PROGRAM USED TO TEACH OUR CAI MODEL

First, students receive an overview of why and how instructional design is done for CAI. Following that, the student learns how to:

1. Determine the objectives of a lesson.

2. Conduct a task analysis of the lesson (hierarchy or info processing)

3. Classify the learning outcome(s) of the lesson

4. Design a Readiness for Learning lesson component

   i. assess prior knowledge of the learner
   ii. recall prior knowledge in learner
   iii. describe objectives & content of lesson

5. Design the CAI instructional component:

   First, by choosing outcome to be taught (several cycles of this may be completed in one lesson, for multiple outcomes).

   Second, by choosing instructional strategy for the given outcome (tutorial, game, etc.).

   Third, determining the necessary microstrategy of instructional events, based on text prescriptions and teacher experience.

   Fourth, following workbook guidelines to construct lesson component on paper.

6. Convert workbook program design into screen displays.

7. Conduct formative evaluation of lesson via screen displays.

EVIDENCE OF THE MODEL'S SUCCESS
(This will be summarized in the presentation)

Results of approach: no hard empirical data: this is tough to derive when evaluating the effectiveness of a semester-long intervention via use of a course text. However, 15 graduate teacher education students in a Spring '87 class used the approach in designing tutorial software. Their instructional products using a Superpilot system were all at the A or B+ level for a project grade (as graded by the authors of this presenation), with appropriate readiness activities, interactivity, and practice sequences.

The biggest student difficulty in the course was in appropriately identifying the proper learning outcomes (confusion on rules vs. concepts, rules vs. problem solving). However, students tended to embody the basic instructional events for each type of outcome in their lesson, and (as we hoped) transcend these with the addition of their own supplementary design components, indicating that teachers may not have to be taught all design components for each CAI strategy/outcome as much as they need to learn the basic microstrategy instructional events from which to work. Students' course ratings were highest on survey questions about the ease of use of the workbook and its help in achieving course goals. Since students programmed their lessons themselves, we were not able to test the premise that a teacher's screen displays can be coded into the computer by another programmer.

OTHER FEATURES OF OUR APPROACH
(Included in AECT presentation if time permits)

1. Explanatory feedback should be provided for right answers, which are more than just confirmation of success. This is because students can guess at right answers, and also make "correct but not perfect" answers that have some mistakes in them (Hazen, 1985). Also, different types of wrong answers receive different types of feedback, depending on the type of wrong answer, and number of tries.

2. Use of learning strategies as an option for tutorial programs. In the workbook, teachers are shown how CAI lessons can ask students to generate their own images, mnemonics, or examples for a given learning outcome.

3. Deemphasis of verbal information CAI lessons. Teachers are told that verbal information lessons can be more efficiently taught outside of CAI. If verbal information is a prerequisite of intellectual skill of the CAI lesson, it can be taught on the computer.

4. Reference-text approach. For the first part of the book, everyone learns the same necessary design activities (task analysis, objectives, outcomes, readiness for learning). For the next part, students can reference individual sections of the book, depending on the outcomes they are trying to teach. In the last section, everyone completes the same section on screen display design. This allows for a learn-as-needed approach to CAI lesson design. With this format, the book can also be used as a reference tool for on-the-job courseware design.

As part of a formal course, teachers can also complete all the workbook exercises in each chapter, to achieve a better understanding all types of CAI instruction. This is explained in a How to Use this

Book section of the workbook, a feature missing from too many instructional texts.

5. 3 Stages of formative evaluation. The book explains that there are three cycles of formative evaluation that can be conducted for CAI: a) evaluation of the front-end analysis of objectives and task analysis by an SME (a teacher or a designer), b) evaluation of the screen displays by students and/or an SME, c) evaluation of the lesson on the computer.

11.

References

Bonner, J. (1987) Computer-based courseware: frame-based or intelligent? Educational Technology, 27 (3), 30-32

Collis, B. & Gore, M. (1987) The Collaborative model for instructional software development. Educational Technology, 27(2), 40-44.

Gagne, R., Wager, W., & Rojas, A. (1981) Planning and authoring computer-assisted instruction lesson. Educational Technology, 21(9), 11-15.

Hazen, M. (1985). Instructional software design principles. Educational Technology, 25 (11), 18-23.

Resnick, L. (1976). Task Analysis in instructional design: some cases from mathematics. In D. Klahr (ed.), Cognition and instruction, Hillsdale, NJ: Erlbaum

Smith, C. (1987). Educators as courseware developers: the key to successful microtechnology integration. Educational Technology, 27(7), 31-33.

Schiffman, S. (1986). Software infusion: using computers to enhance instruction. part one: what does software infusion look like? Educational Technology, 26(6), 7-11.

Tessmer, M., Caverly, D. & Jonassen, D. (in press). A nonprogrammer's guide to the development of microcomputer software Littleton, Colorado: Libraries Unlimited.

Figure 1
Strategy-by-Outcome Matrix for Microstrategies

	Verbal Info	Concepts	Rules	ProblemSolving
Tutorial	MICROSTRATEGY	Definition Proto. Examp. Proto. Nonexp. Examp Classf. Attb. error fdbk.		
Simulation				
Game				
Drill and Practice				
Problem Solving Tools				